ARK Systems

XSCF

SCF Extender File Manager

User's Manual

Quality Software

Printed in U.S.A.

OS-9/680x0 Program

XSCF

User's Manual

Rev. C 1/93

Copyright © 1991, 1992, 1993 ARK Systems USA
All rights reserved.

Printed in U.S.A.

Copyright © 1991, 1992, 1993 ARK Systems USA.

This document contains proprietary information which is protected by copyright. All rights are reserved. Reproduction of this document, in part or whole, by any means, electrical or otherwise, is prohibited, except by written permission from ARK Systems USA.

The information contained in this document is believed to be accurate as of the date of publication, however, ARK Systems USA will not be liable for any damages, including indirect and consequential, from use of the program(s) or reliance upon this documentation.

The information herein is subject to change without notice.

XSCF is a trademark of ARK Systems USA. OS-9 and OS-9/68000 are trademarks of Microware Systems Corporation.

We express special thanks to Mr. Steven Weller of Windsor Systems for his contribution to the improvement of this manual.

This manual reflects XSCF V1.1.

Printing History

Rev. A	11/91	First edition.
Rev. B	2/92	Added compatibility explanations.
Rev. C	1/93	Corrected errors. Added V1.1 changes

Table of Contents

Chapter 1
Introduction

1.1 Preface

XSCF is a file manager level program package running under the OS-9/68000 operating system. This program package contains a new file manager called **XSCF**. XSCF runs on any existing OS-9/68000 systems and reinforces the user interface capabilities to increase your productivity.

XSCF was designed with independence from particular hardware and software in mind. There is no need to make changes to the currently working environment, guaranteeing the complete compatibility with the current system.

1.2 XSCF — SCF Extender

XSCF is an ***SCF Extender*** that extends the line editing function of the Sequential Character File Manager (SCF), one of OS-9's standard file managers responsible for managing terminals, printers, and other serial devices. When used on terminals, SCF provides very limited line editing capabilities, mostly teletype-like operations. This has frustrated many OS-9 users. XSCF provides at the ***file manager's level*** the advanced line editing and line recall capabilities that are seen on other operating

systems.

XSCF's line editing capabilities allow you to update, delete, and insert characters at any position in the current entry line. If your terminal has **arrow** keys and other special keys such as "**home**" and "**end**," you can very easily configure XSCF to exploit those keys.

The line recall capabilities of XSCF are extremely useful when you use several command lines repeatedly: The "***smart recall buffer***" algorithm sorts the lines in the recall buffer such that frequently used lines are always placed at the top of buffer; the "***associative recall***" algorithm retrieves only the lines that match the first characters typed in. These will enable you to type the fewest key strokes to recall desired lines.

Since we think "*look and feel*" are very important for this kind of user interface functions, we have made XSCF as compatible with SCF as possible: All the special keys used by SCF work exactly the same or even more conveniently. You'll get used to XSCF — and become unable to live without it — in a minute!

Another innovative and important feature of XSCF is its interface to the operating system. You don't have to make any changes to your system configuration. Just typing a single command line switches your current terminal path to an XSCF path. No system generation or modifications is required.

XSCF differs greatly from so-called "enhanced shells" and other application programs that offer line editing facilities, by virtue of its implementation as a ***file manager***.

Here is a summary of XSCF's features:

- Extended line editing capability — updates, deletes, inserts characters at any position in the current entry line.
- Smart line recall buffer with associative recalls.
- Completely compatible with SCF.
- Virtually no installation — a single command line makes everything go.

1.3 Notes

The programs included in the XSCF program package run under V2.2 and later versions of OS-9/68000.

Although we have tried to make this manual as easy to understand as possible, it is not intended to be a tutorial manual for the OS-9 operating system. The user of the XSCF program package is assumed to have fair amount of knowledge about OS-9 and its operation principal. Should you feel you don't, please take advantage of seminars held by Microware and other parties, or grab an OS-9 expert at your site.

1.4 Conventions in This Manual

Throughout this manual, the computer display texts are printed `like this`, while the echobacks of the user's key entries are *`italicized`*. Unless otherwise instructed, remember to hit the carriage return key at the end of each line entry.

Chapter 2 Installation

2.1 Distribution Disk

The distribution disk contains the following files (file names in UPPERCASE LETTERS found in the diagram on the next page represent directories):

axscf	The XSCF-attaching program.
bxscf	The XSCF device descriptor browser program.
makekeys	The XSCF device descriptor customization program.
xscfadm... xscfvt100	XSCF device descriptors.
xscfdtemplate . .	A template device descriptors for customization.
read.me	This file may not appear on your copy of distribution disk. If it does, *LIST* it for the latest information about XSCF.

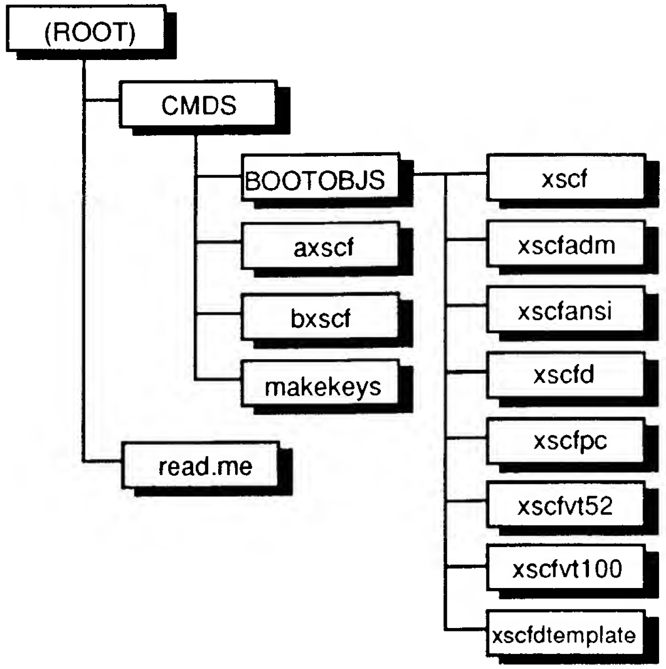

Fig. 2.1 Distribution Disk Contents.

2.2 Installation

Installing XSCF is very easy, just copying the files on the distribution disk is all. First, make sure that your current execution directories is on your system disk (usually hard disk). Typing the following command line makes it certain:

```
$ chx /dd/cmds
```

The command lines assume that your system disk is "**/dd**." If not, substitute the device names in the command lines appropriately.

Second, insert the distribution disk into a floppy disk drive, then

type as follows:

```
$ chd /d0; dsave -ies /dd
```

The above command line assumes that your floppy disk drive in which you have inserted the distribution disk is "**/d0**" and that the system disk is "**/dd**." If not, substitute the device names in the above command line appropriately.

2.3 LoadList File

If your OS-9 system has a line like "`load -z=SYS/LoadList`" in the ***startup*** file, it is a good idea to put the following module names in your ***LoadList*** file:

xscf, xscfadm, xscfansi, xscfd, xscfpc, xscfvt52, xscfvt100
(all files are found in the BOOTOBJS directory)

Nevertheless, this is optional and merely for eliminating typing commands to load these modules individually, and eventually wastes a few hundred bytes of memory. Read the rest of this manual. You will find out which modules really need to be preloaded.

That's all! You're done with the installation. Refer to the next chapter for how to use XSCF.

Chapter 3
Using XSCF

3.1 Choosing Device Descriptor

XSCF is designed to be so flexible that it can work with many types of terminals with different special key codes. Anyway, you have to choose an appropriate XSCF device descriptor. The XSCF program package contains the following several device descriptors for typical terminals:

xscfadm	for terminals with ADM3 type key codes,
xscfansi	for terminals with ANSI type key codes,
xscfpc	for terminals with PC-AT type key codes,
xscfvt52	for terminals with DEC VT-52 type key codes,
xscfvt100	for terminals with DEC VT-100 type key codes,
xscfd	for terminals with no special keys.

In the following explanations, it is assumed that your terminal is compatible with DEC VT-100 and you have chosen the **xscfvt100** device descriptor. If your terminal is not found in the top five of the above list (**xscfadm~xscfvt100**), your terminal has no special keys such as **arrows**, or you are not sure of your terminal's special key codes, choose **xscfd** for the time being and substitute "**xscfvt100**" in the following explanations with "**xscfd**." You can

create your own terminal device descriptor later. Refer to the next chapter for complete terminal key code information.

NOTE: The terminal types listed in the previous page are just hints and may not properly correspond to actual terminal models. Consult your terminal's operating manual and the key code table in the next chapter for complete key code information for each device descriptor; some terminals may have several different "modes" that generate different key code sets. If your terminal's key codes do not agree with those in the table or you are not sure, use the *MAKEKEYS* program to create a custom device descriptor. See the next chapter

The terminal device descriptors supplied in the XSCF program package correspond to particular *terminal types* but not to *port hardware*. All XSCF device descriptors are independent from particular port hardware and therefore can be used with any ports on any OS-9/68000 computers.

3.2 Attaching XSCF

First, load the file manager and device descriptor with the following command line. If you have added the XSCF module names to your load module list (see the previous chapter) and have rebooted the system, skip to the next paragraph. Here, it is assumed that you have chosen the **xscfvt100** device descriptor.

```
    $ load bootobjs/xscf bootobjs/xscfvt100
```

Should the *LOAD* command line result in an error, make sure that you have correctly installed XSCF and your current execution directory is /dd/CMDS.

Then type the following command line:

```
    $ axscf /xscfvt100
```

Upon success, type ^A (type 'A' while holding down the control key). If you see a line as shown below and the cursor is placed at the end of the line, the attaching is completed!

```
    $ XSCF Copyright (c) 1991 ARK Systems USA
```

Type ^X to erase the line, then type any command lines such as *PROCS* or *DIR*. Your terminal works just as usual.

3.3 Using Recall Buffer

After executing several command lines, type ^A. The most recent command line you entered comes back; nothing new so far...

Type ^A again. Now you can tell the difference of XSCF from SCF: the older lines are coming back! Keep typing ^A, XSCF remembers approximately 1,000 characters in its recall buffer.

If you have overtyped ^A, type ^Z to advance the recall pointer. You can go back and forth with these keys to any point in the recall buffer. All keys of XSCF have been chosen so that they are easy to type while keeping compatibility and natural upgrade path from SCF's standard line editing key arrangement.

If your terminal has ***up*** and ***down arrow*** keys, these keys behave the same as the ^A and ^Z keys: **up arrow** recalls back

and **down arrow** advances.

> **NOTE:** The **up** and **down arrow** keys do not work if you have chosen **xscfd**.

Whenever a line is recalled and entered, XSCF relocates the line to the top of the recall buffer so that the recall buffer contains no duplicated lines and that the most recent entry will be recalled most quickly. If the recall buffer becomes full, the oldest entry line(s) will be abandoned.

3.4 Associative Recall

The **up** and **down arrow** keys work *almost* the same as ^A and ^Z, but differently in one respect.

Suppose you have typed several command lines since you copied a file *xscf.r* from /dd/USERS/ARK/PROGS/UTILS/XSCF/RELS to /d0/ARK/SAMPLES/DEMO/IO/XSCF/RELS some time ago and you now want to copy *axscf.r*; we bet you'll really thank XSCF's recall function. But how many times do you have to type ^A until you reach the line? XSCF helps you reduce even more number of key strokes.

In this case, type '**c**' at the beginning of the line then hit the **up arrow** key. Be sure that there are no characters in the entry line before you type the '**c**'. XSCF recalls lines beginning with '**c**' while skipping others. If there could be other lines such as "**code**" which also begin with '**c**' but they are not what you want, type "**cop**" as the template. XSCF looks for matching lines.

The **down arrow** key works oppositely: it advances the recall pointer while skipping unmatching lines. If you type either ^A or ^Z, XSCF leaves the associative mode and the **up** and **down arrow** keys become to behave exactly the same as ^A and ^Z, until another template string is specified.

NOTE: The associative recall function is not usable with the **xscfd** device descriptor.

3.5 Editing Entry Line

One typical kind of frustration with SCF is that, when you have mistyped even a single character at the beginning of a long entry line, you have to go back the character location by erasing many characters and retype them again. XSCF's complete line editing capability will relieve your frustration.

The ***left*** and ***right arrow*** keys move the cursor back and forth respectively in the current entry line.

NOTE: Use ^J and ^L instead of **left** and **right arrows** respectively if you have chosen **xscfd**.

The ***backspace*** key removes the character at the left of the cursor (if any) and shifts all the trailing characters (if any) to the left by one. The ***delete*** key works almost the same as **backspace** but it removes the character at the current cursor location.

The ***PF1*** (or ***home***) and ***PF4*** (or ***end***) keys make the cursor jump to the top and end of the current entry line respectively.

NOTE: Use ^U and ^O instead of **home** and **end** respectively if you have chosen **xscfd**.

The ***PF2*** (or ***insert***) key toggles the insert mode on and off. In insert mode, the characters typed in are inserted at the current cursor position and the characters on and at the right of cursor (if any) are shifted to the right. On some terminals you may see different cursor shapes or blinking speeds for insert and non-insert modes. The default mode is non-insert and hitting carriage return returns to the default non-insert mode.

NOTE: Use ^I instead of **insert** if you have chosen **xscfd**.

No matter what editing keys you have typed and no matter where the cursor currently is, hitting carriage return terminates the line entry operation, and no matter what appears on the entry line is read by the application program.

3.6 Alternative Attaching

If you never leave XSCF, there's a more convenient way to attach XSCF: a pathlist consisting of an XSCF device name followed by an SCF device name automatically opens the SCF device path with the XSCF device attached. For example, if your ***startup*** file contains a line to invoke the time sharing monitor such as "`tsmon /term`," change this to "`tsmon /xscfvt100/term`."

When you reboot the system next time, your terminal is already attached to XSCF at a login. (The necessary modules must have been loaded before *TSMON*.)

You may think it is satisfactory to attach XSCF in the ***startup*** file and start the TSS monitor on the same terminal later; this will not work. Although both the ***startup*** procedure and TSS monitor use the same terminal *device*, they open different *paths*. XSCF is attached to a path, not to a device.

3.7 The AXSCF and BXSCF Commands

The *AXSCF* command provides the means to control and monitor an XSCF path. If *AXSCF* is invoked without an XSCF device descriptor name, it displays the key codes assigned in the current XSCF path. If *AXSCF* is invoked with an XSCF device descriptor, it attaches XSCF to the current standard input path.

AXSCF accepts the following options:

`-a`	Enter auto-insertion mode: when the cursor is on a space or slash character, regular character entries are inserted rather than overwritten.
`-b`	Exit auto-insertion mode.
`-c`	Display escape sequences in character format.
`-d=`*delay*	Set delay to wait for a complete key code sequence. This prevents missing key code bytes from a slow terminal or low baud rate communication line. XSCF waits at least *delay* ticks or one tick.
`-n=`*nulcnts*	Ignore *nulcnts* consecutive null bytes.
`-m=`*minchars*	Set minimum string length to put into the recall buffer to *minchars*. The default is 3 characters.
`-w=`*path#*	Operate with the path of *path#*. The default is the standard input path (*path#* = 0). This is necessary if *AXSCF* is used in a shell procedure file.

The *BXSCF* command browses XSCF device descriptors whose file names are given in the command line. By default, the current execution directory is searched. Below are valid *BXSCF* options.

`-c`	Display escape sequences in character format.
`-d`	Search files from current data directory.
`-m`	Search memory module directory.
`-z[=`*file*`]`	Read file/module names from standard input or *file*.

Chapter 4
Keys and Key Codes

4.1 Device Descriptors

XSCF retrieves special key code information from the path descriptor's option area, which is copied from the device descriptor when the path is opened. Thus, different terminal types that generate different key code set must use different XSCF devices.

The XSCF program package contains several device descriptors that are suitable for popular terminal types. The special keys and their key codes used in those device descriptors are listed in the next page. If your terminal generates a different key code set for special keys, use the *MAKKEYS* customization program. which is described in the next section. In the table, "`$**`" represents a hexadecimal number and Esc the ASCII escape code (`$1b`).

XSCF uses the same line-editing keys as SCF, such as ^A for line recall and ^X for line delete. These key codes are found in the SCF path descriptor attached to an XSCF path, rather than in the XSCF path descriptor. So, these line editing keys can be modified with *TMODE* as well as usual SCF paths. Currently no way is provided to change XSCF's special key codes other than re-opening the XSCF path on a different XSCF device descriptor.

device	**xscfadm**	**xscfansi**	**xscfpc**	**xscfvt52**	**xscfvt100**	**xscfd**
recall back	↑	↑	↑	↑	↑	^A
	`$1e`	Esc O A	`$00` H	Esc A	Esc [A	`$01`
recall forward	↓	↓	↓	↓	↓	^Z
	`$1f`	Esc O B	`$00` P	Esc B	Esc [B	`$1a`
cursor right	→	→	→	→	→	^L
	`$1c`	Esc O C	`$00` M	Esc C	Esc [C	`$0c`
cursor left	←	←	←	←	←	^J
	`$1d`	Esc O D	`$00` K	Esc D	Esc [D	`$0a`
cursor top	shift←	***prev***	***home***	***PF1***	***PF1***	^U
	`$02`	Esc [5 ~	`$00` G	Esc P	Esc O P	`$15`
cursor tail	shift→	***next***	***end***	***PF4***	***PF4***	^O
	`$06`	Esc [6 ~	`$00` O	Esc S	Esc O S	`$1f`
insert	***ins***	***insert***	***insert***	***PF2***	***PF2***	^I
	`$12`	Esc [2 ~	`$00` R	Esc Q	Esc O Q	`$09`
delete	***del***	***remove***	***delete***	***delete***	***delete***	***del***
	`$7f`	Esc [3 ~	`$00` S	`$7f`	`$7f`	`$7f`

Table 4.1 Keys and key codes used by XSCF.

4.2 Customizing Device Descriptor

If your terminal generates different key codes from those listed above, use *MAKEKEYS* supplied in the XSCF program package to create your own custom device descriptor. First, type as follows:

```
    $ makekeys -m
```

MAKEKEYS first asks for the new device descriptor's name. The new device name must be less than 12 characters in length. Type in any valid module name you like; however, we recommend using a name with "**xscf**" followed by your terminal's model name.

Next, you are asked to press the special keys used by XSCF. If your terminal doesn't have the key(s), press ***carriage return*** or ***enter*** instead; those keys simply will not function. *MAKEKEYS* echoes the key codes back to the terminal.

Here is an example:

```
$ makekeys -m
**************************************************
              "makekeys" edition #7
         Copyright 1991 ARK Systems USA
         Creaing XSCF device descriptor.
**************************************************
Enter device name: xscfark
Press requested keys.
If your keyboard doesn't have the key(s), press Enter or
Return instead.
      [Up Arrow]  > $fes
    [Down Arrow]  > $fer
   [Right Arrow]  > $fep
    [Left Arrow]  > $feq
        [Insert]  > ^L
        [Delete]  > $7f
   [Top of line]  > ^B
   [End of line]  > ^F
Max delay: 0 tick.
Ignore null count (0-255)? 0
Can your terminal change cursor shape or speed? (y/n) y
Enter sequence to set normal mode cursor: $1b.$0e$0f
Enter sequence to set insert mode cursor: $1b.$00$0e
Are you sure to create the device descriptor module? (y/n) y
Creation of XSCF device descriptor completed.
```

Each special key may have up to 5 (five) bytes. The "ignore null count" sets the number of consecutive null characters ignored by XSCF. This is merely for old terminals that send null characters after carriage return for synchronization. For most modern terminals this is unnecessary, so enter zero or simply hit carriage

return. If your terminal can change cursor shape or speed with appropriate escape sequences, answer 'y' to the question. Refer to your terminal's operating manual for the escape sequences. When entering escape sequences, use a hexadecimal notation preceded by '$' for all non-printing characters and the dollar character itself. The program accepts characters until carriage return is hit, but at most 9 (nine) character bytes are usable.

MAKEKEYS reads a template device descriptor **xscfdtemplate** from the BOOTOBJS directory under the current execution directory (usually /dd/CMDS) and creates the new device descriptor file in the same directory (BOOTOBJS). Be sure to put the new device descriptor name in your ***LoadList*** file to load the module automatically at a start up. **xscfdtemplate** has the same key codes as **xscfd** so that the keys that were not found on your terminal have the same key codes as **xscfd**.

> **NOTE:** To use XSCF's line editing capabilities fully, ***left arrow*** (pure backup) must be distinguished from ***backspace*** (destructive backup). Some terminals such as Qume QVT-102, Wyse WY-50, and Televideo 925 map the same key code to both ***backspace*** and ***left arrow***. We recommend using the **xscfd** device descriptor for such terminals.

If your terminal is relatively slow or connected by a low baud rate line, you may see XSCF not correctly interpret multiple byte key code sequences. Should this happen, try *AXSCF*'s -d option to increase delay time. The delay time is counted in ticks and is put only when XSCF detects a possible first byte in a sequence; so it does not slow down normal key entries. *MAKEKEYS* automatically checks your terminal's speed and puts a rough value in the device descriptor, but *AXSCF*'s -d option overrides it.

Chapter 5
Advanced Topics

5.1 Compatibility

Although XSCF is compatible with most programs that access SCF paths: it performs completely transparently to application programs. A few programs, however, need special considerations as follows:

- **SrcDbg** — Source Level Debugger — cannot use XSCF in a simple manner: you need to set up some trick. A dedicated **SrcDbg** launcher program "*LSRCDBG*" makes **SrcDbg** exploit XSCF fully. *LSRCDBG* is sold separately and available from ARK Systems USA; call one of our authorized distributors.
- **Debug** — Assembly Code Debugger — currently cannot use XSCF (but its child shell and descendant processes can) because **Debug** does not use the standard paths but opens its own to the SCF device instead.
- **SysDbg** — System Debugger — and **ROMDebug**, both for debugging device drivers and other system level programs, cannot use XSCF.
- XSCF may coexist with so-called **"enhanced shell"** programs that have their own line editing and recall capabilities, if they use **I$Read** to read key input. Nevertheless, you cannot take

advantage of XSCF unless you can turn off the line editing and recall capabilities of such programs.

- Some application programs may be "too smart" about path name validity checking: an XSCF pathlist is considered illegal because it has a second path name element which is not allowed in an SCF path. This problem occurs only when the application is to open a new XSCF path, and thus very rarely; however, if this happens, use *"ADFM"* — Alias Device File Manager — on top of XSCF. *ADFM* is sold separately and is available from ARK; call one of our authorized distributors.
- The **VJE** Kanji entry system for the Japanese language environment cannot use XSCF because the device driver and the *knj* module perform all line editing functions on behalf of the file manager.

5.2 Other Considerations

- XSCF is usable only with SCF device paths.
- Upon attaching, XSCF overrides the SCF path, but the SCF path remains open and provides XSCF with the means to access the physical device.
- Since XSCF is attached to SCF *paths*, all processes that share the same path by inheritance or duplication are affected while paths that are separately opened on the same SCF *device* are not affected.
- XSCF intercepts the **I$ReadLn**, **I$GetStt**, **I$SetStt**, and **I$Close** system calls to the attached SCF path. All other I/O system calls are directly passed to SCF and therefore work absolutely identically to a regular SCF path.
- The *TMODE* command correctly works as if the path were a normal SCF path, but *PROCS* reveals the real path device.

- Currently there's no way to detach XSCF from the SCF path. If you really want to stop using XSCF, close the path completely (including all duplicated paths by inheritance in process fork) and reopen the SCF path. If you are on a time sharing system, a logout will completely close your terminal path.
- XSCF uses the **null** device driver supplied with the standard OS-9 package, but never accesses it. The port base address of an XSCF device descriptor must be set different from "/**pipe**," but can be the same as ARK's PSCF PostScript file manager.

5.3 Theory of Operation

XSCF implements the line editing capabilities by positioning itself between the kernel and the SCF file manager, as shown below:

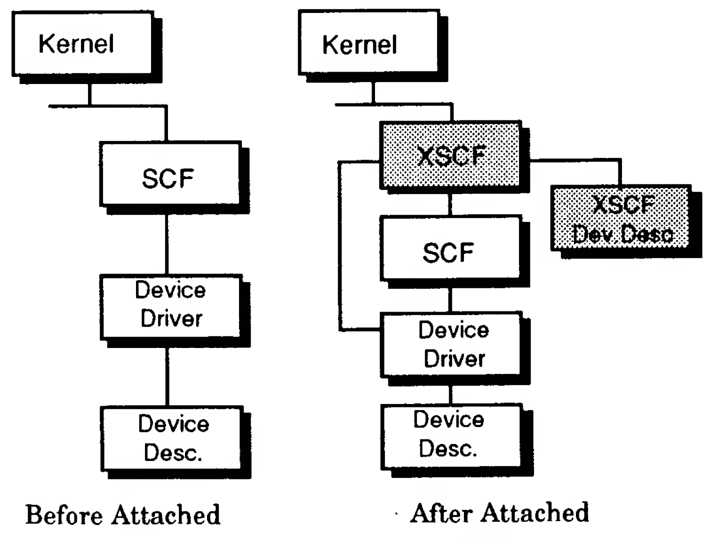

Fig. 5.1 Attaching XSCF.

There are two ways to attach XSCF to an SCF path: if you want to open a new XSCF path on an XSCF device, specify a pathlist whose first element is an XSCF device name and second element an SCF device name, XSCF internally opens a path to the device and attach the path to XSCF; if you want to attach XSCF to an SCF path that is already open, use the *AXSCF* command which manipulates the path descriptor table and the path descriptors. Attaching XSCF in different ways results in no difference at all.

5.4 More Key Codes

XSCF is actually capable of handling more special keys than described in the previous chapter: *Page Up/Down*, *Skip to Right/Left Word*, and *Delete Right/Left Word*(*Page Up/Down* are currently not used). If your terminal's arrow keys can generate different key codes when pressed with the shift or control key, these keys will be useful. Invoke *MAKEKEYS* with the `-ma` option; it will ask for more keys.